Semantics-Based Virtual Product Models: Unifying Knowledge and Product Data

Mike Uschold, Sean Callahan
Boeing Phantom Works
P.O. Box 3707,m/s 7L-40
Seattle, WA USA
98124-2207
{michael.f.uschold, sean.callahan}@boeing.com

Abstract

We are concerned with the goal of applying semantics-based technologies to enhance product development capability, including data, processes and tools, to make it faster and cheaper to design and deliver new products whose fundamental behaviors, and failure modes are well understood and predictable. The main contributions of this paper are: 1) the presentation of a conceptual framework for understanding this goal and 2) setting a research agenda for achieving this goal. We start with an introduction to semantics-based technologies focusing on ontologies, and the strongly related areas of knowledge representation, and automated inference. We present some common high-level uses for ontologies, and consider how they apply to virtual product models.

One important requirement to achieve our goal is the formulation of precise and disciplined approach to capturing a virtual product representation for a complex product like the space shuttle, or a commercial jet liner. Towards this end, we will highlight some key distinctions that can be used to differentiate between the many kinds of information that are variously and ambiguously referred to using terms such as: 'knowledge', 'information', 'data', 'product design representation,' etc. We conclude by identifying some items that can form the basis of a research agenda in this area.

1 Introduction and Motivation

The goal of this paper is to identify how and whether various semantics-based technologies may be used to help meet the wide variety of challenges that we face in moving towards the goal of virtual product modeling. Chief among these is the requirement for successful management of the wide variety of information relating to products. In what follows, we use the term 'product information' in a non-technical way to refer to anything [that can be stored and retrieved] relating to design, representation and reasoning about products. Successful information management means getting the right information to the right people at the right time. This, in turn, gives rise to further requirements. We require a precise and disciplined approach to capturing a virtual product representation for a complex product like the space shuttle, or a commercial jet liner. To facilitate efficient access to the information, we require a shared and reusable framework for structuring and organizing the wide variety of product information. It is necessary to accommodate the widely varying formats and conceptual models with some mechanism to ensure interoperability. Increasingly, there is a need for automated reasoning to infer implicitly specified information (e.g. for forensics), and to ensure logical consistency in design specifications. In broad terms, it is necessary for the software systems supporting the product development lifecycle to be maintainable in cost-effective way. Finally, there is a need to distinguish between data representations that are strictly descriptive from those that require inference or approximation. We believe that clearly articulating these distinctions will allow a clearer focus on distinct, important research area critical to making progress on virtual product modeling.

2 Semantics-Based Technologies

Motivated by the above concerns/requirements, we introduce semantics-based technologies, with the aim of identifying which of the above requirements can be addressed by these technologies.

2.1 Introduction to Ontologies, KR and Automated Inference

To meet a variety of needs in information modeling, software development and integration as well as knowledge management and reuse, various groups within industry, academia, and government have been developing and deploying sharable and reusable models known as ontologies. Much of the material from this section is drawn from [Bradshaw *et al* 04], additional material on the emerging field of ontologies may be found in [Staab & Studer, 04].

2.1.1 What is an ontology?

The most commonly quoted definition of an ontology is *"a formal, explicit specification of a shared conceptualization"* (Gruber 1993a). A *conceptualization*, in this context, refers to an abstract model of how people think about things in the world, usually restricted to a particular subject area. An *explicit specification* means that the concepts and relationships of the abstract model are given explicit names and definitions. The name is a term, and the definition is a specification of how the term necessarily relates to other terms. *Formal* means that the specification is encoded in a language whose formal properties are well understood—in practice, this almost always means logic-based languages. Formality is an important way to remove ambiguity that is prevalent in natural language; it also opens the door for automated machine processing of intended meaning. *Shared* means that the main purpose of an ontology is generally to be used and reused across different applications and communities.

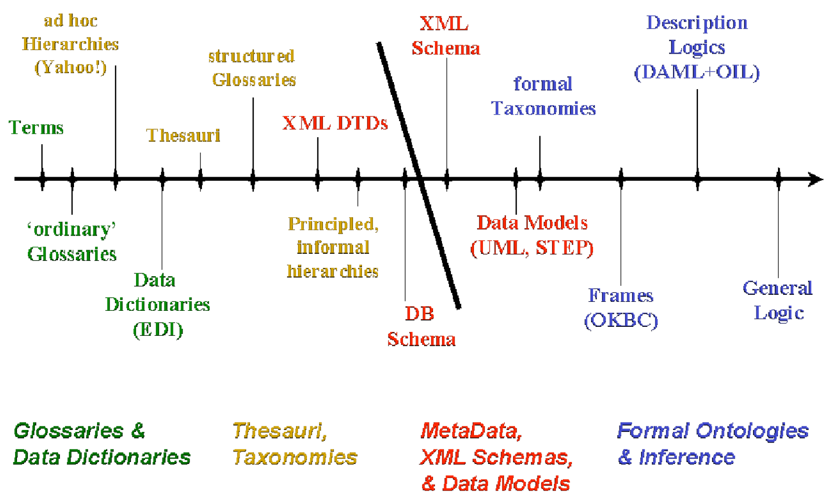


Figure 1: Kinds of Ontologies

While there is much disagreement about the details of exactly what is or is not an ontology, the common thread that runs through virtually all approaches to ontologies, is that there are two essential components of any ontology that is suitable for automated reasoning:

- a vocabulary of terms that refer to the things of interest in a given domain,
- some specification of meaning for the terms, grounded in some form of logic.

What distinguishes different approaches to ontologies is the degree and manner of specifying the necessary relationships among terms. This gives rise to a kind of continuum of kinds of ontologies. At one extreme, we have very lightweight ones that may consist of terms only, with little or no specification of relationships among the terms (the degenerate case of an ontology). At the other end of the spectrum, we have rigorously formalized logical theories, which comprise the ontologies (figure 1). As we move along the continuum, the number of necessary relationships specified increases (thus reducing ambiguity), the degree of formality increases, and there is increasing support for automated reasoning.

2.1.2 Ontologies vs. Knowledge Representation and Inference

Ontologies are a key part of a broader range of semantics-based technologies which include the areas of knowledge representation and automated inference that arose within the Artificial Intelligence (AI) community. Knowledge representation has been mainstream in AI for a few decades. Fundamentally, the aim is to represent and reason about knowledge in the world. Many different representation formalisms have been explored, and reasoning engines developed. A key result is the proven existence of tradeoff between representational power (i.e. the ability to represent/express many different kinds of knowledge) and the efficiency of the reasoning engines. A surprising discovery was made, that a seemingly innocuous minor addition to a language can result in a major loss in inference efficiency. In some cases, it is provably impossible to develop an inference engine that can be guaranteed to give an answer. You may never know whether in the next second it will find an answer, or whether it will go on computing indefinitely and never find one. It is for this reason that the W3C initiative producing a standard ontology language for the Semantic Web produced 3 versions: OWL-Lite, OWL-DL, and OWL-Full[1]. Each is progressively more expressive, and correspondingly, less efficient for reasoning.

In one sense, ontologies are a sub-area within knowledge representation. This is especially true for formal ontologies. Traditional KR languages are used for representing formal ontologies and standard inference engines are used for reasoning over ontologies. Also, a good knowledge base will frequently have an ontology as its backbone. A major difference is that, unlike for KR in general, the key focus of ontologies is on knowledge *sharing*.

In another sense, ontologies go outside the bounds of KR. For example, many so-called 'lightweight-ontologies' have little relevance to KR (e.g. the Yahoo! taxonomy is often called an ontology). The ontologies community may also be viewed as a major customer for KR. In this sense, the use of ontologies is really applied KR.

2.1.3 What does an ontology consist of?

Each kind of thing (e.g. wing, physical object) represented in an ontology (generally called a *class* or *concept*) is typically associated with various *properties* (also called *slots* or *roles*) describing its features and attributes as well as various restrictions on them (sometimes called *facets* or *role restrictions*). An ontology together with a set of concrete *instances* (also called *individuals*) of the class constitutes a knowledge base [Noy & McGuinness 2001] . Classes, arranged in a taxonomy of greater generality at the top and more specificity toward the bottom, are generally the focus of most ontologies. See Figure 2 for an example of an ontology that organizes its terminology as a taxonomy.

[1] See http://www.w3.org/2004/01/sws-pressrelease for a press release about the OWL language family.

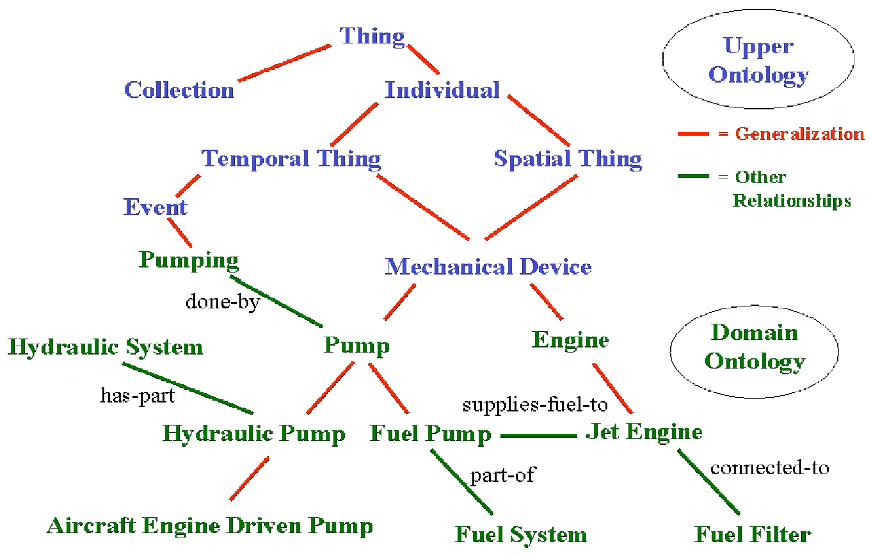
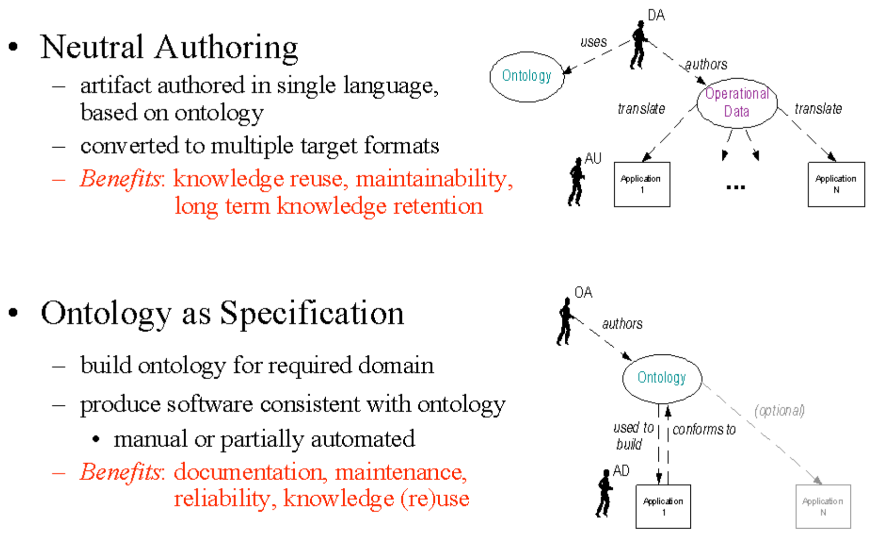


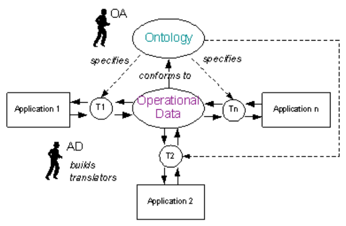
Figure 2: An Example Ontology



- Neutral Authoring
  - artifact authored in single language, based on ontology
  - converted to multiple target formats
  - *Benefits*: knowledge reuse, maintainability, long term knowledge retention
- Ontology as Specification
  - build ontology for required domain
  - produce software consistent with ontology
    - manual or partially automated
  - *Benefits*: documentation, maintenance, reliability, knowledge (re)use

Figure 3: Ontology Application Scenarios

- Common Access to Information
  - information required by multiple agents
  - expressed in wrong terms/format
  - ontology used as agreed standard, basis for converting/mapping
  - *Benefits*: interoperability, more effective use/reuse of knowledge



- Ontology-Based Search
  - Ontology used for concept-based structuring of information in a repository
  - *Benefits*: better information access

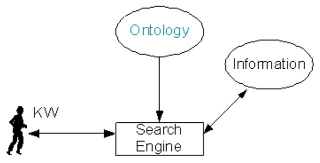


Figure 4: More Ontology Application Scenarios

2.1.4 How do ontologies help?

The promise of ontologies is "*a shared and common understanding of a domain that can be communicated between people and application systems*" (Fensel 2001). Ontologies may be applied in a number of ways. Some of the most important applications are given below (Jasper and Uschold 1999). The key aspects of these classes of applications are summarized in figures 3 and 4.

Neutral Authoring. "An information artifact is authored in a single language and is converted into a different form for use in multiple target systems. Benefits of this approach include knowledge reuse, improved maintainability, and long-term knowledge retention."

Ontology as Specification. "An ontology of a given domain is created and used as a basis for specification and development of some software. Benefits of this approach include documentation, maintenance, reliability, and knowledge (re)use."

Common Access to Information. "Information is required by one or more persons or computer applications, but is expressed using unfamiliar vocabulary or in an inaccessible format. The ontology helps render the information intelligible by providing a shared understanding of the terms or by mapping between sets of terms. Benefits of this approach include interoperability and more effective use and reuse of knowledge resources."

Ontology-Based Search. To facilitate search, an ontology is used as a structuring device for an information repository (e.g., documents, web pages, names of experts). The chief benefit of this approach is faster access to important information resources, which leads to more effective use and reuse of knowledge resources.

2.2 Application to Virtual Product Models

In [Uschold *et al,* 1999], we discuss some approaches to knowledge sharing using ontologies that we tried at Boeing in various contexts. Here, we discuss in more general terms, which of the above ontology application scenarios may be exploited in the context of virtual product modeling, and how.

Neutral Authoring: Given the plethora of non-interoperable tools and formats, a given company or organization can benefit greatly by developing their own 'neutral format' for authoring, and then develop translators from this format to the formats required by the various target systems. This would entail a very time-consuming effort in developing the neutral format (aka ontology) such that it has adequate coverage in the subject matter of interest. To ensure no loss in translation, the neutral format must include only those features that are supported in *all* of the target systems. The tradeoff here is loss of functionality of some of the tools – many of the special features may not be usable.

From the point of view of consumers of product design software systems, it would be a great leap forward if there was a standard that all the vendors were required to support. Unfortunately, market forces constitute a great barrier to this happening. Vendors currently compete on the extra functionality that they provide. The market place would have to change so that vendors instead competed on speed, efficiency and convenience of supporting a standard format with a fixed functionality. This is analogous to compiler vendors for standard programming languages.

Common Access to Information: There are many different high level views of representing product information, as well as many different low-level formats. It will always be necessary to translate between these different formats and representations. While it is safe to assume there will not be global ontologies and formats agreed by one and all, it is nevertheless possible to create an ontology to be used as a neutral interchange format for translating among various formats. This avoids the need to create and maintain O(N**2) translators. It also makes it very easy for new systems and formats to interoperate with many others.

This scenario is similar to neutral authoring, in that a neutral format is created in the domain of the tools that must interoperate. One difference is that the translation is two way rather than one way. Interoperability between any two systems happens by first translating from the source system format into the neutral format, and from there into the target system format. Another difference is in how the neutral format is designed. While it is inevitable for some loss of translation to occur between systems with differing functionality, it is desirable to limit this by ensuring that there is no loss of translation from the source format to the neutral format. Thus, the neutral format must cover all of the things that occur in *any* of the target systems. That is, in terms of expressiveness, the neutral format is the union rather than the intersection of the expressiveness of the target systems. With this approach, it would also be a great advantage to have a global standard format. Market forces work against this approach also. Vendors do not want to provide translations to allow customers to use other tools.

Ontology-based Specification: There is tremendous scope for developing a large-scale 'product ontology' which is used as the basis for developing a wide variety of software. There is a similarity here with the neutral authoring scenario, in that an ontology that was developed as a neutral format could also be used as the basis for more general product software development. The same ontology can be used as the basis for multiple systems. The difference is the relationship between the ontology and the target system. For neutral authoring, there is a translator built to convert specifications in the neutral format into the required target system formats. Other than this, there need not be any special relationship between the target software systems and the neutral format. In the case of ontology-based specification, the ontology is used as the basis for software development. Indeed, this is what you would have if authoring systems were specifically developed for entering specifications that would be output to the neutral format. The whole GUI and usage mode would be driven by the ontology, which in effect serves as a specification for what can be created by end users of the product design software.

It goes much further than product design software, however. A whole range of product support software could also be driven from the same ontology – including viewing and presentation tools, data bases, and even marketing and accounting tools that are used to track product sales. This would ensure much easier interoperability among these software systems whose relationships are typically only implicit. The link between the ontology and the software may be informal, much as an informal design requirements document is used to write software. As the software evolves, the ontology used as a basis for specifying the software becomes out of date. A much better approach is to have a formal link between the ontology and the software. This is the approach of the so called "Model-Driven Architecture" created and promoted by the OMG[2]. There are also some ontology software vendors which automatically create Java classes and Java Documents from an ontology[3]. When the ontology changes, the code is automatically updated. A large variety of applications may use the accessor functions from the ontology. Not only does this ensure greater interoperation, but it also offers significant cost

[2] http://www.omg.org/mda/
[3] See, for example: http://www.ontologyworks.com/PR-patent.htm

reduction for software evolution and maintenance. There is a growing interest in the idea of “Ontology-Driven Software Engineering”[4] which is precisely what this ontology application scenario is about.

Ontology-based Search: The final major category of uses for ontologies is for organizing, classifying and understanding product design information, at a higher level of abstraction than is commonly used today. They can be used as a sophisticated indexing mechanism into product structures. A very familiar example of this use of ontologies is the Yahoo! subject taxonomy, which can be thought of as a lightweight ontology. It is well suited to its use for browsing the Web. However, close examination of this taxonomy reveals that it is somewhat ‘semantically challenged’ – e.g. it is not a strict taxonomy, the links have many different meanings. This is fine for a human, but makes it impossible for a machine to do automated reasoning with predictable results. In the much more specialized virtual product context, there are advantages of having a semantically richer and cleaner ontology.

This completes our brief examination of how ontologies may be used in a virtual product context, based on the general categories noted above. These application scenarios are *not* mutually exclusive. It is specifically intended that these different approaches be combined in interesting ways to achieve greater benefits than any one approach could on its own. For example, the same ontology may be used to structure a repository of product information as well as serve as the basis for achieving interoperability through neutral authoring or as an interchange format. The same ontology could also be used as the underlying basis for a suite of product-related software. Next we consider the other major theme of this paper: ‘product information’ in much greater detail. .

3 Unifying Knowledge and Product Data

In this section, we consider the myriad of different kinds of data, information and knowledge relating to design, representation and reasoning about products. This, in conjunction with the previous section, will provide the basis for identifying a research agenda for moving towards the goal of applying semantics technologies to enhance the functionality of today’s product support software

The overall goal is to have a product development capability, including data, processes and tools, that makes it faster and cheaper to design and deliver new products whose fundamental behaviors, and failure modes are well understood and predictable. Everything we are addressing here is aimed at what can we do to move ourselves along a vector toward that goal. Our underlying thesis is that achieving this requires substantial automation of basic design processes, and also much more rigorous virtual product modeling technologies, including simulations of desired functionality, and simulations of the designs created to deliver the functionality. The creation, modification and interpretation of this design data require the application of judgment and knowledge.

The literature is full of references to ‘product knowledge’ as compared say to product data, or product information. However, there is no shared view of what ‘product knowledge’ is or isn’t. The different views expressed in the literature at best will be talking about similar things, but using very different language. More commonly, they are talking about different things. Rarely is there a clear attempt to precisely characterize what is meant by the term ‘product knowledge’. Worse, the views are sometimes contradictory. This makes for great confusion, and holds the community back from systematically addressing the design automation and virtual product modeling issues.

Our goals in this section is to offer a particular set of categories into which product design related information can be classified to lend a clarity to the research and development tasks that will lead to more design automation and better virtual modeling capabilities. Note that we specifically are *not* suggesting what the term ‘product knowledge’ should or should not be used for – no one would pay attention anyway. We do believe that the following general information categories will prove useful, if only as a basis for getting the issues out there and starting a discussion. We will first name them and describe them briefly, and then provide a more general discussion, with examples, to illustrate the ideas.

[4] See, for example, http://www.bpiresearch.com/WP_BPMOntology.pdf

1. **Precise Product Data (PPD)** – Unambiguous information representations that can be used to capture specific designs, and whose purposes are to describe those particular designs and not to make general statements about whole classes of designs. Such representations include those which capture the designs of a product families, each of which include a product architecture and many distinct designs variations, or product family members.

2. **Precise Extraction Information (PEI)** – Information that can be used to extract additional precise information from PPD. We consider this to be equivalent to capturing PPD, since it is a bit like a lossless compression algorithm. System architects can trade off between PPD and PEI depending upon implementation considerations.

3. **Imprecise Extraction Information (IEI)** – Information used to extract additional data from PPD using imprecise or heuristic inference and assumptions. This extracted information was known (or knowable) at the time the PPD was created, but just wasn't captured. This data is of inherently less quality than PPD / PEI, and it seems like it should be reduced or eliminated as we become more sophisticated about product design capture.

4. **Product Augmentation Information** – This is computer encodable information needed to interpret or understand a product definition, but is applicable to many different product designs. This data may be of a precise or imprecise nature. An example is part type hierarchy information, on one end of the spectrum, or rules for deriving and updating the finite element mesh for a complex structural part, on the other. Informal or unstructured information like textual rules of thumb that can't be computed upon would not be in this category, but instead reside in category 5.

5. **Other Knowledge** – This is a catch all category to pick up what is not covered in the first four. This is a combination of poorly characterized or understood subjects that may eventually move into one of the other categories as our understanding increases, and subjects that will fundamentally remain the purview of human gray matter. Whether the information theoretically relegated to gray matter is a zero or nonzero set is a matter of philosophical concern that will be left unaddressed here.

We believe that understanding and categorizing product design related information relative to these categories will provide a clearer research focus in the areas of design automation and virtual product modeling and authoring. The very idea that, by increasing our level of understanding, we can move information from one category to another allows sophisticated design-system implementation tradeoffs to be undertaken in a systematic manner. In the remainder of this section we will discuss each of the categories in more detail with emphasis on examples to illustrate what is intended.

The definition of Precise Product Data (PPD) is relatively straight-forward and current representations are appropriate to their purpose. Semantics-based technologies will not supplant them, but rather enhance or augment their use. In a sense categories 2, 3, and 4 all very naturally represented using the wide array of semantic technologies that are available. As we proceed with the category discussions we will intersperse examples of how those technologies are employed and how they relate.

Definition of Precise Product Data (PPD): The principle is that PPD are the results of design activities and of decisions regarding what to include and exclude in a product's function, system definition, form and manufacturing plans. While the knowledge of categories 4 and 5 are clearly used in the production of such data artifacts, what is left at the end of that process is a concrete quantity, analogous to the capital value in a completed building or of a milling machine ready to cut metal or be sold for its capital value. While one can say that such artifacts are available as models, and so indirectly contain knowledge about a design, we would argue that the job of extracting that type of general information is exactly what categories 4 and 5 provide. Consistent with that, PPD does not include things like design rational, or general classification hierarchies for parts or systems. A non exhaustive list of some PPD would include the following:

- Bills of materials and product structures

- Detailed systems designs including system-subsystem decompositions and schematic diagrams
- CAD part and assembly data
- Manufacturing assembly sequences and simulations
- Accountability mappings between design domains (e.g. logical system design to engineering CAD design)
- Requirements
- Functions
- Product simulations

An interesting case to consider is that of parametric CAD data, where well parameterized models can be used to create a whole class of designs. We would consider the resolved model with a specific set of parameters to be PPD. The information used to determine the parameter set in a given context would be Product Augmentation Information or General Knowledge. The CAD system, which reads in the parametric model, calculates the constraints and produces new models based on particular parameter inputs, is clearly Product Augmentation Data.

For a specific example of trade-off between PPD and IEI consider the design representation used to capture a hydraulic system. If the connectivity of the hydraulic system is not directly captured anywhere, then you might need to build a sophisticated knowledge capture/mining application using Imprecise Extraction Information (IEI) to traverse the geometric design of the system deducing connectivity, implying which parts are tubes by their aspect ratios and, generally speaking, re-deriving the schematic diagram abstraction of the hydraulic system. If, on the other hand, the schematic diagram is captured as part of the product data, and further linked to the physical design of the system, then there is no need for any extra tools with embedded knowledge about how to extract the needed information. If complete clues were embedded in the incomplete product data, it may be possible to extract the connectivity information exactly, in which case the extraction information would be Precise Extraction Information (PEI). An example of such information is capturing the part type and subsystem type that each part is e.g. whether a part is a tube.

The main relevance of this line of thinking to ontological / semantic technology is twofold. First, it can allow us to reduce their application to cases that will benefit them most. Second, it can provide clarity regarding the areas in which to focus semantic and knowledge based technologies to get maximum benefit.

In some sense, a very useful and focused research agenda can involve deciding whether different types of product related information and analysis belong in the PPD category or in the Other Knowledge category. That which goes into the PPD category, but which is not yet realized in current product representation models can act as a guide for where to expand it, or in the case of product level simulation models it lays out a fundamental research agenda whose purpose is to develop a coherent theory and practice of product level simulation which can then be a basis for enriching current PPD models. In areas such as complex product simulation, having overlapping research attempting to extend PPD and ‘Other Knowledge’ in such areas is important and necessary, for both the long and short terms.

On the other hand, research, like much of the geometric feature recognition research has been much less useful. Basically, boundary representation solid modeling loses much of the understanding of the designed part’s shape that was in the engineer’s head at design time, and so a whole research industry sprung up around trying to extract that data which was lost at design time. To be fair, manufacturing feature decompositions whose purpose its to couple with specific machine tool path planning activities was not in the engineers head at design time, but the general geometric feature recognition problem is generally too hard (or maybe ill-posed), and so for both cases not many useful results have been produced, especially in proportional to the efforts expended. Given this, one must really wonder how much more progress would have been made in this area if that same effort had gone toward helping people capture relevant information at authoring time, in the form of PPD.

3.1 Examples of Increasing Precise Product Data

Over the last several years specific progress has been made toward enriching the current scope and coverage of PPD representations. [Callahan/Heisserman 1997], and have formalized simple reusable design information patterns, and applied them to multiple design domains for one product design in [Callahan 1997]. A port connectivity formalism was added to that basic reusable design paradigm in [Heisserman/Mattikalli 1999]. In [Callahan 2002], [Callahan 2003], [Callahan 2004] all of this was generalized to address these same data domains, but for a product family in which the members share a substantial fraction of their underlying design data.

The latter product family extensions were done to address two key, competing requirements driving the underlying product design information models that are needed to achieve the greater levels of product design completeness necessary to support virtual design environments, for complex product families: The first requirement is to capture a much greater range of product data, spanning from design requirements and functional behavior specifications to manufacturing plans for specific product variations. The second requirement is, to the greatest degree possible, to share this more complete design data between all members of a product family. Maintaining common design content is very difficult if the underlying design data is not shared, and the current state-of-the-art does not effectively support either the sharing of design data between product family members, or capturing integrated design data spanning the full scope of product design.

A data representation / information infrastructure that can simultaneously meet both of these requirements by extending the concept of reusable designs to reusable design architectures is presented in [Callahan 02, 03 & 04]. It is shown that capturing the design data for a complex product family in terms of its architecture and its configurations achieves maximum design data sharing, and representational scalability in terms of object count and intelligent user interface support. Within the design of a product family there are multiple reusable design architectures, one for each distinct design domain, like detailed systems design and engineering CAD / assembly design. Adding accountability mappings between related domains creates a full-scope product architecture.

We now recap the enrichments of PPD as described above:

1. Formalize reusable design concepts.
2. Support a systematic way to map between multiple design hierarchies within one design (e.g. system design to engineering CAD design).
3. Formalize connectivity / relationship capture in a reusable design hierarchy
4. Generalize 1, 2, 3 to allow each design hierarchy to represent a family of designs, and efficiently map between these generalized reusable design structures (called Extended Generic Product Structures).

Each one of these steps removed substantial needs to extract information using knowledge based or semantic ontologies as illustrated for each step, below.

1. Allowed assembly design to be defined once and reused many times, so that duplicate assembly definitions did not have to be compared to determine whether they were the same. A simple example of this is to apply a transform to an unchanged part to relocate it, as opposed to using the transform to modify all of the internal part geometry to be defined fundamentally in the new location.
2. Informal mappings between system designs and their CAD implementations were originally tracked by human beings looking at a schematic and at a 2D drawing and trying to make sure all of the system entities were capturing in the mechanical design. With these extensions, those mappings are captured explicitly and a simple report or interactive application can show what they are.
3. Instead of using geometric comparison to determine 3D connectivity for a system, the representation captures what should be connected to what, and a simple geometric test can verify each connectivity point and report an error if there is not geometric contact.
4. Instead of complex algorithms to compare assemblies that have evolved from a common starting point, this representation encodes those similarities and differences explicitly and with a novel user interface simply reports to the end-user what they are.

We believe that substantial progress was made above due to the fact that we distinguished between the underlying data needed to capture a design versus data that was needed to reason about and augment that data. It is instructive to note that all of this PPD extension was done because we were trying to automate the creation of airplane designs (with an initial focus on hydraulic systems), and facilitate the integration of airplane systems simulations with the CAD design of the airplane.

3.2 Relationship between PPD and Neutral Representation Ontologies

Before concluding, we would like to make one final point that we feel is a good stepping off point for starting a discussion that might help develop these ideas. As described in section 2, a very important area, from a practical viewpoint, is that of design data exchange between parties in a distributed enterprise. Those parties may use different CAD tools, product data management systems or other different design software tools that cover overlapping data scopes. When this happens you often get two (not entirely consistent) views of what PPD is or should be.

This job of mapping between PPDs feels very much like the extraction categories 2 and 3 in section 3. Section 2 describes a common approach of developing a third PPD and calling it an ontology. A good question, which we don't have a complete understanding yet, is whether that is the best way to pose the problem, or whether it might be better to just view it as another PPD, and to concentrate semantic technologies on providing the precise or imprecise mappings between multiple partially or completely compatible PPDs.

4 Summary and Conclusions

We have introduced the area of semantics technologies, focusing on ontologies and the strongly related areas of knowledge representation and automated inference. We identified some general categories for applying ontologies, in general, and then gave some specific suggestions as to how these uses may apply for virtual product modeling. We then shifted gears, and proposed several specific categories of product related data, information and knowledge, and noted some of their key inter-relationships.

The two main sections of the paper are as yet, only weakly connected, reflecting what we understand to be the state of the art. The overall goal of setting a research agenda is to find out how better to integrate our understanding of semantics technologies, and to apply it to virtual product modeling. We identify the following areas that can serve as the basis for further research:

1. Create a comprehensive framework for understanding and classifying the many facets of product information. In this paper, we have merely begun to lay the foundation for this – more complete coverage and systematic analysis is needed. This framework would aim to reflect the usage of terms that are common today, and suggest some distinctions and categories for the community to agree on.
2. Test the above ideas about how ontologies and semantics technologies may be applied to the area of virtual product modeling. The above suggestions are very general, specific projects must be undertaken. They would identify specific scenarios for applying semantics technologies in the areas mentioned and carry out feasibility studies producing prototype implementations that test and demonstrate the ideas.
3. Identify a comprehensive set of requirements for product modeling, that can be used as a source of inspiration for how/whether semantics technologies may apply. It may also serve as a checklist, of a sort.
4. Have ongoing workshops which bring together experts in both product information/modeling as well as in semantics technologies.

References

[Bradshaw *et al* 2004] Semantic Integration. In Bradshaw, J. M., Boy, G., Durfee, E., Gruninger, M., Hexmoor, H., Suri, N., Tambe, M., Uschold, M., & Vitek, J. (Ed.). (2004). *Software Agents for the Warfighter. ITAC Consortium Report.* Cambridge, MA: AAAI Press/The MIT Press (to appear)

[Callahan 1998] "Relating Functional Schematics to Hierarchical Mechanical Assemblies," 1998, Third Annual conference on *Computer Aided Design*

[Callahan 2002] US Patent Pending – serial number 10/128,922. A LOGICAL HIERARCHICAL DATA MODEL FOR SHARING PRODUCT INFORMATION ACROSS PRODUCT FAMILIES

[Callahan 2003] US Patent Pending – serial number 10/348,470. APPARATUS AND METHOD FOR MANAGING MULTIVARIANT ASSEMBLY DATA MODELS

[Callahan 2004] Extended Generic Product Structure**:** Maximum Information with Minimum Data: submitted to ASME Journal JCISE (Journal of Computing and Information Science in Engineering)

[Callahan – Heisserman 1997] A product representation to support design automation. In M. Pratt, R. D. Sriram, and M. J. Wozny, editors, Product Modeling for Computer Integrated Design and Manufacture (Proc. 5th IFIP WG5.2 Workshop on Geometric Modeling, Airlie, VA, 19-23 May 1996) Chapman & Hall, London, 1997.

[Fensel 2001] Fensel, D. . *Ontologies: Dynamic Networks of Formally Represented Meaning* http://www.cs.vu.nl/~dieter/ontologies.pdf (paper) http://www.cs.vu.nl/~dieter/ftp/slides/ibrow.05.2001.pdf (slide presentation)

[Gruber 1993] Gruber, T.R. 1993a. A translation approach to portable ontology specifications. *Knowledge Acquisition* 5:199-220.

[Gruninger & Uschold 2004] Chapter in Agents book by Bradshaw

[Heisserman – Mattikalli 1998] Representing Relationships in Hierarchical Assemblies. Proceedings of DETC '98, 1998 ASME Design Engineering Technical Conferences, September 13-16, 1998, Atlanta Georgia, USA

[Jasper & Uschold 1999] Jasper, R. and Uschold, M. 1999. A Framework for Understanding and Classifying Ontology Applications. In Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW'99,. http://sern.ucalgary.ca/KSI/KAW/KAW99/

[Noy & McGuinness 2001] Noy, N. F. &McGuinness, D. L .*Ontology Development 101: A Guide to Creating Your First Ontology*. Knowledge Systems Laboratory, Technical Report KSL-01-05, March, 2001. http://ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html.

[Staab & Studer 04] *Handbook on Ontologies*; Springer Series on Handbooks in Information Systems
S. Staab, R. Studer (eds.)

[Szykman & Sriram 2001] "The Role of Knowledge in Next-generation Product Development Systems" in the *Journal of Computing and Information Science in Engineering;* Transactions of the ASME, March 2001, Vol. 1.

[Uschold *et al,* 1999] Uschold, M., Jasper, R. and Clark, P. 1999. Three Approaches for Knowledge Sharing: A Comparative Analysis. In *Proc. 12th Workshop on Knowledge Acquisition, Modeling, and Management* (KAW'99).

[Uschold & Gruninger 1996] Uschold, M. and Gruninger, M. 1996. Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 1:96-137.